Anna LISIAK, Piotr LECH

MIKRO-KLASTRY OBLICZENIOWE W ROLI KOMPUTERA POKŁADOWEGO

Streszczenie

W artykule przedstawiono dwa modele mikro-klastrów obliczeniowych realizujących zadania komputera pokładowego pojazdu. Rozpatrzono dwie koncepcje – pierwszą realizującą model ukierunkowany na realizację rozproszonych usług, drugą związaną ze skalowalną mocą obliczeniową systemu. Podano potencjalny zakres zastosowań oraz podano, jakie ryzyko niesie ze sobą wdrażanie systemów klastrowych w pojazdach. Niewątpliwą zaletą stosowania mikro klastrów jest ich skalowalność i unifikacja, pozwalająca na dopasowanie oferowanych opcji systemu do modelu biznesowego poprzez optymalizacje kosztów produkcji. W ramach testów zestawiono przykładowy klaster, składający się z pięciu mikrokomputerów o identycznej konfiguracji wraz z dedykowanym interfejsami wyjściowymi. Interfejsy wyjściowe miały za zadanie generowanie obciążeń obliczeniowych dla całego systemu klastra oraz wizualizację działania testowych algorytmów.

WSTĘP

Komputery pokładowe pojazdów szynowych, samolotów czy też samochodów muszą spełniać wysokie normy jakościowe i niezawodnościowe. Współczesne urządzenia pokładowe powinny posiadać moc obliczeniową, którą wykorzystają nowoczesne algorytmy sterowania oraz inne oprogramowanie systemowe. Dynamiczny rozwój technik programistycznych, które gwarantują szybkie wdrożenia, wiążą się ze znacznym zapotrzebowaniem na moc obliczeniową oraz są wymagające, co do zasobów takich jak pamięć masowa, pamięć RAM itp. Zamknięte systemy opracowane jednorazowo dla realizacji konkretnego zadania często przy zmianie oprogramowania, po wprowadzeniu nowych funkcji stają się mniej wydajne, co może mieć wpływ na stabilność i bezpieczeństwo systemów. Rozwiązaniem powyższego problemu może być zastosowanie skalowanych systemów, zwiększających swoje możliwości wraz z rozwojem aplikacji lub wprowadzaniem nowych usług. Pojawienie się na rynku modułów mikroprocesorowych realizujących zadania system on chip SoC – daje możliwość tworzenia systemów klastrów obliczeniowych lub usługowych. Mikro klaster obliczeniowy jest odpowiednikiem rozwiązań znanych z dużych ośrodków obliczeniowych i wykorzystuje podobne technologie do ich budowy. Mikro klaser jest w pełni skalowalną platformą obliczeniową, zwiększającą swoje możliwości obliczeniowe poprzez dodawanie kolejnych modułów mikroprocesorowych. Istnieje możliwość wprowadzenia nadmiarowości do systemu klastra (który jest zbudowany z większej ilości jednostek niż jest wymagane) w celu zapewnienia większej niezawodności systemu. Nadmiarowe jednostki w przypadku uszkodzenia pojedynczej (lub kilku) jednostki przejmą jej zadania, ważne jest zapewnić wystarczający zapas mocy obliczeniowej w innych nadmiarowych jednostkach. Alternatywą dla klastra obliczeniowego jest klaster usługowy, znany i stosowany w technologach sieciowych do zapewnienia wydajnego udostępniania usług sieciowych w sieci komputerowej. Klasycznym przykładem klastra usługowego jest system obsługujący sieć WWW. Obecnie najbardziej rozpowszechnione są systemy realizowane w systemach otwartych Linux lub w technologiach zamkniętych np. wspieranych przez Microsoft. Systemy z modułami SoC posiadają bogate wsparcie społeczności Linuxowej, co znacznie przyśpiesza rozwój tych systemów, a ich popularność przekłada się na ceny. Duża ilość aktywnych developerów i testerów modułów powoduje, że te technologie są stabilne i niezawodne. Podstawową cechą klastra jest jego niewidoczność oraz jego ukrycie. W praktyce urządzenia peryferyjne nie wiedzą, że współpracują z klastrem, identycznie jak w przypadku pracy z pojedynczym komputerem pokładowym. Jednostki mikroprocesorowe tworzące klaster są częścią całości. Aplikacje nie powinny widzieć różnicy między pracą z pojedynczym komputerem pokładowym lub klastrem. Wskazane jest, aby system posiadał cechy klastra niezawodnościowego HA (High Availability). - o wysokiej dostępności. W razie uszkodzenia jednego z mikrokomputerów jego zadania są, w sposób niewidoczny dla aplikacji i systemów, przejmowane przez inny mikrokomputer należący do klastra.

1. ARCHITEKTURA SPRZĘTOWA

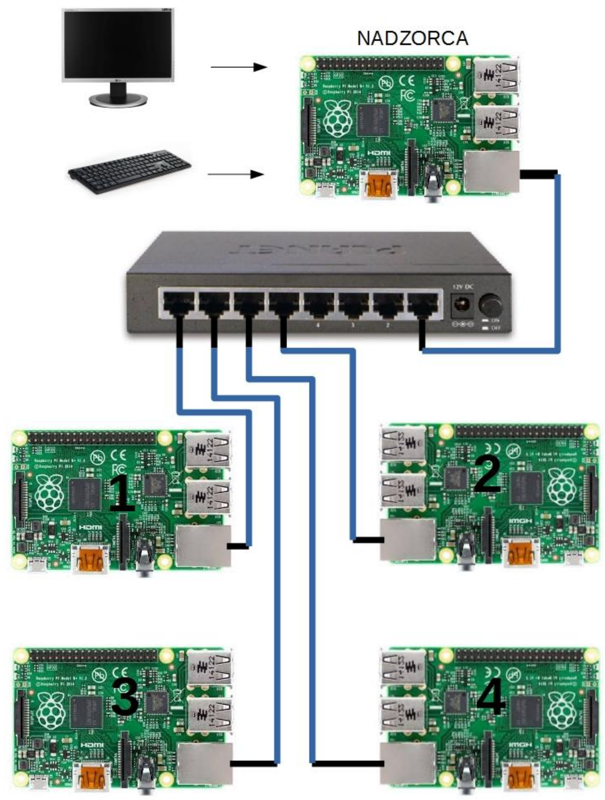
Popularny mikrokomputer Raspberry Pi (w skrócie RasPi) został wybrany jako podstawowa jednostka obliczeniowa. Klaster wykorzystuje pięć jednostek obliczeniowych, komunikację zapewnia sieć komputerowa w standardzie IEEE802.3. Wszystkie jednostki uczestniczące w eksperymencie posiadały zainstalowaną najnowszą dystrybucję Linuxa - raspbiana - będącego klonem dystrybucji Debiana na platformę Raspberry Pi. Przyjęto założenie, że jedno rozwiązanie techniczne – prototyp mikro-klastra - będzie w stanie w zależności od konfiguracji pracować jako klaster usługowy albo obliczeniowy.

Z całej rodziny dostępnych mikrokomputerów RasPi został wybrany model o dobrym stosunku ceny do możliwości, o następujących parametrach:
- procesor Broadcom SoC BCM2835 700 MHz z zintegrowanym układem graficznym
- pamięć RAM - 512 MB
- pamięć sytemu operacyjnego SD 8GB
- sieć w standardzie IEEE802.3
- cztery porty USB
- 40 portów GPIO
- rozmiar: 86 x 56 mm.

Jedynym mankamentem przyjętego rozwiązania jest brak możliwości pracy z systemami czasu rzeczywistego, przez co nie nadają się do sterowania procesami czasowo krytycznymi. System komputera pokładowego można zrealizować w oparciu o proponowaną platformę sprzętową zakładając, że procesy czasowo krytyczne obsłużą inne dedykowane konkretnym zadaniom sterowania regula-

tory. Klaster jedynie posłuży jako urządzenie przekazujące nastawy do regulatora oraz jako odbiornik komunikatów. Istnieje jednak duża ilość zagadnień, które system ten może obsłużyć np. ograniczonego sterowania nadrzędnego, powiadomień głosowych, multimediów, wizualizacji stanu urządzeń itp.

Istotą klasteryzacji jest stworzenie pojedynczego super komputera złożonego z wielu jednostek. W tym celu komputery zostały spięte w sieć komputerową za pomocą ośmio-portowego switcha firmy Planet. Wszystkie komputery pracują w tej samej sieci IP, parametry sieci są statyczne i niezmienne w czasie. Z pięciu jednostek wydzielono jedną i przeznaczono na nadzorcę systemu. Podstawowa konfiguracja pozostałych jednostek była identyczna. Tak zestawione komputery stworzyły system, który zależnie od konfiguracji może pracować w jednym z założonych modeli pracy klastra - obliczeniowego lub usługowego. Rysunek 1 przedstawia schemat blokowy urządzenia a rysunek 2 widok zmontowanego prototypu.

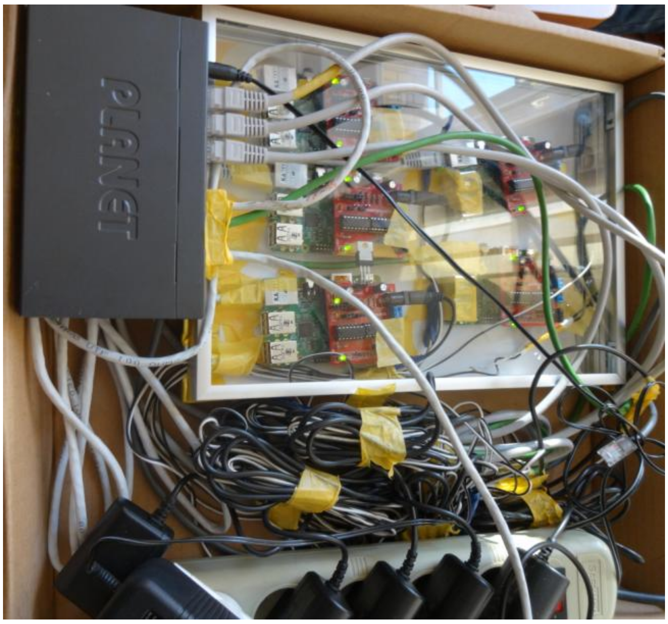


Rys. 1. *Schemat budowy mikro-klastra*

Szczególne znaczenie ma wydzielona jednostka, która stanowi węzeł komunikacyjny nadzorujący system tzw. zarządca jest to jedyny w systemie mikrokomputer RasPi, który umożliwia podłączenie klawiatury i monitora. Pozostałe jednostki mogą być konfigurowane zdalnie poprzez terminal znakowy lub w wygodniejszy sposób poprzez webowe narzędzie administracyjne Webmin. Zaletą oprogramowania Webmin (rysunek 3) jest przyjazny interfejs graficzny i wsparcie technologiczne do zarządzania klastrem. W przypadku połączenia z siecią zewnętrzną Webmin umożliwia zarządzenie z dowolnego komputera wyposażonego w przeglądarkę internetową.

2. MIKRO-KLASTER USŁUGOWY

Klaster usługowy ma za zadanie optymizować dwa aspekty – pierwszy związany z ruchem informacji w systemie – drugi związany z zapotrzebowaniem na moc obliczeniową. Aspekt związany z ruchem jest rozpatrywany w systemach, w których posiadamy kilka krytycznych dla systemów źródeł dużej ilości komunikatów, których pojedyncza jednostka nie jest w stanie przetworzyć i poprawnie reagować na napływające informacje.



Rys. 2. *Zdjęcie prototypu mikro-klastra*

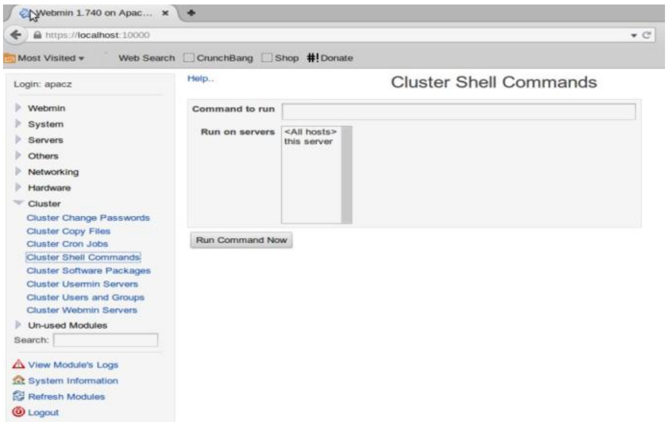
Przykładowo odpowiednikiem tej sytuacji w świecie usług internetowych jest sieć WWW, gdzie strumień zapytań generowanych przez oprogramowanie klienckie (przeglądarki internetowe) obsługują serwery WWW widoczne z zewnątrz jako pojedyncza usługa. W rzeczywistości w węźle systemu następuje rozdział zapytań (wywołań usługi) na poszczególne jednostki klastra dostarczając zapytanie do oczekującej na nie usługi. W tego typu sytuacji strumień informacji rozdzielany jest na kilka jednostek realizujących identyczną usługę. Drugi aspekt przewiduje, że strumień informacji nie jest jednorodny i wymaga przetwarzania przez różne usługi. Nawiązując przez analogię do usług internetowych można sobie wyobrazić stan, w którym kilka usług oczekuje na komunikaty z sieci generujących obciążenie systemu. Oprogramowanie działające w węźle ma za zadanie analizować przychodzącą informację i rozdzielać komunikaty tak, aby możliwe było ich obsłużenie.

Wyróżniamy dwie strategie

- pierwszą, w której badamy stan obciążenia jednostki i w momencie, w którym osiągnie założony stan krytyczny zadania przekazywane są do obsługi przez oprogramowanie kolejnej jednostki; wadą tego typu rozwiązania jest nierównomierne obciążenie jednostek, co daje w konsekwencji szybsze zużycie w pełni obciążonych
- drugą, w której rozkładamy obciążenie na wszystkie jednostki równomiernie, co wpływa na zwiększenie trwałości sprzętu.

Drugie rozwiązanie sprawuje się znacznie lepiej w przypadkach awaryjnych, w których następuje odłączenie jednej z jednostek. W zasadzie prezentowany w pracy prototyp w tej części eksperymentu realizuje zadania klastra równoważące obciążenie (Load Balancing Cluster). Tego typu rozwiązanie często spotyka się w klastrach serwerowych, rozpraszających zapytania pochodzące z programów użytkowników (klienckich). Przeznaczone są do utrzymywania silnie obciążonych usług sieciowych lub prostych zadań obliczeniowych. Ich głównym zadaniem jest równoważenie obciążenia między poszczególne serwery (mikrokomputery). Load balancer przesyła nadchodzące żądania obsługi do mikrokomputerów, które posiadają

wolne w danej chwili moce obliczeniowe, odciążając tym samym inne. W przygotowaniu eksperymentu wzorowano się na projekcie [1]. Testowy projekt do konfiguracji i administracji używał oprogramowania Webmin w odróżnieniu od pierwotnego projektu, który był zarządzany z poziomu zdalnej konsoli.



Rys. 3. *Webmin – aplikacja webowa zarządzająca poszczególnymi jednostkami*

3. BADANIE MIKRO-KLASTRA USŁUGOWEGO

Test mikro-klastra w wariancie klastra usługowego o równomiernym obciążeniu przeprowadzono w oparciu o oprogramowanie zarządzające węzłem i programem realizującym obsługę testowej usługi. Do testów wykorzystano aplikację webową, która miała za zadanie wygenerowanie po stronie serwera odpowiedzi na zapytanie z klienta (skrypt przeglądarki internetowej) o wartość funkcji silnia. Wywołanie tej funkcji było realizowane w nieskończonej pętli. Oprogramowanie klienckie uruchomione było wielokrotnie (wiele instancji w tym samym czasie) w celu wygenerowania znacznej ilości zapytań.

Tab. 1. *Testy wydajnościowe (wyrażone w postaci % obciążenia procesora) mikro-klastra usługowego*

Jednostka	wszystkie jednostki sprawne	awaria 1 jednostki	awaria 2 jednostek	awaria 3 jednostek
1	52%	69%	98%	100%
2	49%	78%	99%	-
3	43%	57%	-	-
4	52%	-	-	-
zarządca	40%	38%	35%	10%

W tabeli 1 przedstawiono wyniki eksperymentu polegającego na fizycznym odłączaniu jednostek w trakcie pracy. Generowało to upadek procesu obsługi pracującego zadania i utratę bieżących wyników. Jednak po około 30ms urządzenie będące węzłem zaktualizowało stan posiadania i kolejne zapytania (zadania) przekazywało do mniejszej ilości jednostek. Niestety w przypadku awarii trzech mikrokomputerów nastąpił stan krytyczny systemu polegająca na nasyceniu jedynej działającej jednostki i brakiem odpowiedzi sytemu. Jeśli typu tego komputer pokładowy ma charakteryzować się dużą niezawodnością należy bardzo dokładnie zbadać stany, w których może dojść do przeciążenia mikroprocesorów i doprowadzić do utraty stabilności.

4. MIKRO-KLASTER OBLICZENIOWY

Klaster wydajnościowy HPC (High Performance Computing) – charakteryzujący się dużą wydajności obliczeniową wykonany został w oparciu o technologie Beowulf [2]. Technologia Beowulf często jest używana do przetwarzania równoległego, poprawiając wydajność systemu poprzez umożliwienie działania na wielu mikrokomputerach jednocześnie. W proponowanym układzie komputery stają się pojedynczym wieloprocesorowym i wielowątkowym superkomputerem. Warunkiem poprawnego działania oprogramowania w tego typu urządzeniach jest zrównoleglenie pracującego algorytmu. Wiele języków programowania wspiera budowę aplikacji, w której procesy składające się na program działają równolegle. Przykładowo język skryptowy python otrzymał wsparcie w postaci dodatkowych bibliotek [4][5].

5. BADANIE MIKRO-KLASTRA OBLICZENIOWEGO

Oprogramowanie testowe zostało napisane w języku skryptowym python. Do badań wykorzystano uproszczoną metodę tworzenia aplikacji wieloprocesowej [6][7][8][9]. Procesy powoływane są do życia na poszczególnych mikrokomputerach cyklicznie a wyniki działania gromadzone i prezentowane w procesie nadzorcy. Ważnym elementem jest sprawdzanie przed uruchomieniem nowego zrównoleglonego wątku czy jednostka jest wolna, działa czy też odłączona. W stosunku do kodu podanego na rysunku 4 zamiast komendy *qstat* została użyta *pstat*. Zaprezentowany kod pokazuje pewną ideę, którą można rozbudować o bardzo zaawansowaną logikę.

```
#!/usr/bin/env python

import re
import subprocess

    command_to_run = "qstat -q"

    p = subprocess.Popen(command_to_run,
stdout=subprocess.PIPE, stderr=subprocess.PIPE, shell=True)
        output, error = p.communicate()
    match = re.search(r'command not found', error)
    if match:
        """something...
    match = re.search(r'Queue', output)
    if match:
        """something else...
```

Rys. 4. *Fragment kodu nadzorcy aplikacji ilustrujący sposób sprawdzenia czy dany mikrokomputer jest aktywny (źródło [5])*

Przygotowano dwa testy :

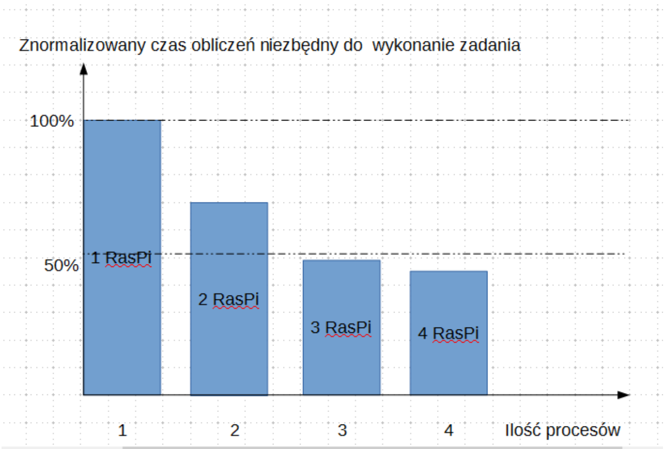
1. realizujący zrównoleglony algorytm mnożenia macierzy,
2. wykorzystujący upubliczniony kod algorytmu estymacji funkcji gęstości jądra w benchmarkingu metodą Parzen–Rosenblatt [6] również napisany w języku python.

W tabeli Tab.2 zestawiono wyniki testów związanych z algorytmem mnożenia macierzy. Widoczne jest szybsze niż w poprzednim usługowym modelu wejście systemu w nasycenie. Wynika to z faktu, że algorytm mnożenia macierzy jest bardziej obciążający system niż algorytm obliczenia funkcji silnia. Z tego samego powodu drugi z algorytmów od początku pracował na granicy możliwości sytemu.

Tab. 2. *Testy wydajnościowe(wyrażone w postaci % obciążenia procesora) mikro-klastra obliczeniowego*

Jednostka	Wszystkie jednostki sprawne	awaria 1 jednostki	awaria 2 jednostek
1	80%	99%	100%
2	76%	98%	-
3	78%	96%	-
4	81%	-	-
zarządca	60%	60%	60%

Z rysunku 5 wynika, że dysponując czterema jednordzeniowymi mikrokomputerami można uzyskać znaczny wzrost wydajności. Jednak nie jest on liniowy. Wzrost wydajności najbardziej zauważalny jest po uruchomieniu dwóch lub trzech procesów. Dodanie czwartego nie jest już tak spektakularne, ponieważ wyraźnie zauważalna jest rola systemu operacyjnego i pracujących innych aplikacji.



Rys. 5. *Wpływ ilości aktywnych procesów na czas wykonania zadania*

PODSUMOWANIE

Mikro-klaster zbudowany z mikrokomputerów Rasberry Pi znacznie zwiększa wydajność obliczeniową. Badania wykazały również, że rozwiązania prezentowane w pracy w ograniczonym zakresie dla małych obciążeń posiadają cechy klastra niezawodnościowego. Uwzględniając powyższe fakty, mikro-klaster może być ciekawą alternatywą dla komputerów pokładowych w systemach transportowych. Powyższe badania powinny być powtórzone i zrealizowane z wykorzystaniem mikro-klastra przystosowanego do pracy w czasie rzeczywistym.

BIBLIOGRAFIA

3. http://raspberrywebserver.com/raspberrypicluster/raspberry-pi-cluster.html; 01.09.2015
4. http://rpiai.com/2015/04/18/python-beowulf-cluster-find-a-use-for-those-computers-that-are-10-years-old/ ; 01.09.2015
5. https://wiki.python.org/moin/ParallelProcessing;01.09.1015
6. http://www.parallelpython.com/;01.09.2015
7. http://stackoverflow.com/questions/29264708/how-to-tell-in-python-if-i-am-running-on-a-beowulf-cluster; 01.09.2015
8. http://sebastianraschka.com/Articles/2014_multiprocessing_intro.html 01.09.2015
9. Lawrence T., Beowulf Cluster Computing With Linux by Sterling 2001 ISBN 0262692740 MIT Press
10. Wilkinson B., Allen M.: Parallel Programming, Prentice Hall, 1999.
11. Bourke T.: Wyrównywanie obciążenia serwerów. O'Reilly, Warszawa, 2002.

MICRO COMPUTING CLUSTERS IN THE ROLE OF THE ON - BOARD COMPUTER

Abstract

The paper presents two models of micro-computing clusters performing tasks of on-board car computer. Two concepts have been considered - the first carries out a model dedicated to the implementation of distributed services, the second is associated with a scalable cluster's computing power. The potential range of applications and that the risks of implementation of cluster systems in vehicles, is identified. An important advantage of the micro-clusters usage is their scalability and unification that allows you to adjust the options, offered to the business model of the system, through optimization of production costs. In part of the tests exemplary cluster consisting of five microcomputers an identical configuration with a dedicated output interfaces has been built. Output Interfaces were aimed at generating computational load for the entire system and visualization of the test operation algorithms.

Autorzy:
Anna Lisiak – Zachodniopomorski Uniwersytet Technologiczny w Szczecinie, Wydział Elektryczny, Studenckie Koło Naukowe Teleinformatyki „Apacz 500";
70-313 Szczecin, ul. Gen. Władysława Sikorskiego 37; Tel: +48 91 449-53-11, Fax: +48 91 449-53-47; E-mail: gk27319@zut.edu.pl
dr inż. **Piotr Lech** – Zachodniopomorski Uniwersytet Technologiczny w Szczecinie, Wydział Elektryczny, Katedra Przetwarzania Sygnałów i Inżynierii Multimedialnej;
70-313 Szczecin, ul. Gen. Władysława Sikorskiego 37; Tel: +48 91 449-53-11, Fax: +48 91 449-53-47; E-mail: piotr.lech@zut.edu.pl